

APPLICATION
FOR
UNITED STATES LETTERS PATENT

TITLE: LOAD BALANCING AND FAULT TOLERANCE FOR
SERVER-BASED SOFTWARE APPLICATIONS

APPLICANT: JAMES R. TRETHEWEY

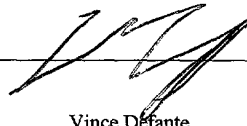
CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EL589643515US

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail Post Office to Addressee with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, Washington, D.C. 20231.

September 18, 2001
Date of Deposit

Signature



Vince Defante
Typed or Printed Name of Person Signing Certificate

10559-494001-P11786

LOAD BALANCING and FAULT TOLERANCE FOR
SERVER-BASED SOFTWARE APPLICATIONS

TECHNICAL FIELD

This invention relates to server-based software applications that involve latency-sensitive message traffic over a packet-switched network such as the Internet.

BACKGROUND

There are many server-based software applications that involve latency-sensitive message traffic over a packet-switched network such as the Internet. In many of these applications, the volume of message traffic is very high. Examples of these applications are Internet telephony, single- and multi-player gaming, music sharing, and other "peer-to-peer" type applications.

An application service provider may make the software application available to many remote users. To accommodate an increase in demand for this service, service providers sometimes add more servers with the same software application to the service provider's network of servers, or server farm. Typically, each of the parallel servers has a unique network address (e.g., Internet Protocol (IP) address). In some instances, service providers install a device known as a load balancer at the "front-end" of the server farm. The load

balancer typically has a unique network address, which is known and used by all clients to access the provider's services. By accessing the load balancer, users need only know one network address, that of the load balancer, to gain access to the service.

The load balancer may distribute incoming traffic among the parallel servers evenly. The load balancer may also direct requests to a specific server if necessary. A load balancer has an upper bound to the amount of traffic it can handle. For example, a typical load balancer may saturate at a rate of 800 requests per second where the data packets are relatively small in size, i.e., light-weight packets. The number of requests that a load balancer can handle is drastically reduced as the size of the data packets increases, i.e., become heavier. Heavy packets may reduce the saturation level of a load balancer to 400 requests per second. For software applications where there is a high volume of traffic from users to the server, the load balancer may become a bottleneck for message traffic. This may lead to an increase in latency. Also, directing all messages through a load balancer in some cases results in an inefficient use of resources.

DESCRIPTION OF DRAWINGS

FIG. 1 is a block diagram of a network configuration that may incorporate aspects of the invention.

FIG. 2 is a flowchart showing the process of establishing
5 communications between a remote computer and a server shown in FIG. 1.

FIG. 3 is a diagram of a probe request data packet used in the process shown in FIG. 2.

FIG. 4 is a diagram of a probe response data packet used in
10 the process shown in FIG. 2.

FIG. 5 is a flowchart showing the process of monitoring the communications between a remote computer and a server shown in
FIG. 1.

Like reference symbols in the various drawings indicate
15 like elements.

DETAILED DESCRIPTION

The invention provides load balancing among similarly functioning software applications residing on parallel servers, while avoiding the problem of a load-balancing device becoming a
20 bottleneck for message traffic from users to the servers. The invention also provides fault tolerance methods for these types of server-based software applications.

In FIG. 1, an Application Service Provider (ASP) 20 has multiple parallel servers 35a, 35b and 35c, each with a unique

network address, for example, unique IP addresses in the case of the Internet. Each server 35a-c is capable of running, for remote users, a software application such as Internet telephony, single- and multi-player gaming, music sharing, and other "peer to peer" type applications. The ASP 20 also includes a load balancer 25, which too has a unique network address, and a switch 30. Together, the load balancer 25 and switch 30 balance the load among the servers 35a-c.

Various computers shown in FIG. 1, namely, computers L_1 - L_6 and N_1 - N_6 , may have access to the application provided by the servers 35a-c. These computers are referred to generally as remote computers because they are remote from the servers 35a-35c. The remote computers have a processor 1 and memory 3. These remote computers have program instructions stored in memory 3 that, when executed by processor 1, transmit requests for a service session to ASP 20 to use a particular software application. In addition, some functions of the software application may be distributed and performed locally on the remote computers. Performing some functions of the software on the remote computers reduces the message traffic between the remote computer and a particular servers 35a-c.

There are two types of remote computers in the embodiment of FIG. 1, local client computers L_1 - L_6 and non-local client computers N_1 - N_6 . Local client computers refer to client

computers that access the group of servers 35 at local ASP 20.

The non-local client computers refer to computers whose local access is to a different group of servers 35 at ASP 21 or ASP

23, or which although non-local computers N_1-N_6 may bypass

5 servers 35 at their local ASP 21 or 23 and access servers 35 at

ASP 20 via Internet 40. The non-local computers N_1-N_6 and ASP 21

or ASP 23 are referred to as non-local networks. The non-local

computers and ASPs 21 and 23 may or may not be in the same local area as local computers and ASP 20. For example, ASP 20 could

10 be located in California, ASP 21 in Minnesota, and ASP 23 in New York. Clients of the respective ASPs may establish

communications with the ASP in their own local area. To

communicate with clients in other local areas, the local ASP may establish communications with a non-local ASP over a packet-

15 switched network such as the Internet 40.

As is conventional, the remote computers may be configured

behind a gateway. FIG. 1 shows remote computers L_1-L_6 configured

behind gateway network address translators NAT_1 15a and NAT_2 15b,

so that each remote computer need not have a unique IP address.

20 The gateway 15a or 15b may perform network address translation,

act as a firewall, or perform both of these functions. Network

address translation is a protocol that enables IP computers in a

private realm to exchange data packets with IP computers in the

public realm.

NAT₁ 15a may be a broadband router located at the user's premises 5. This is typical of a user with a cable Internet access or a digital subscriber line connection. NAT₂ 15b may be a function of a dial-up access device operated at the client's Internet Service Provider (ISP) 10. Through their respective NAT's 15a and 15b, remote computers L₁-L₆ connect to ASP 20 for service.

ASP 20 may have several servers. In this example, there are three servers 35a, 35b, and 35c. These servers each have a unique network address (for example, an IP address) and are configured in parallel, which means that each server has a similarly functioning software application. Each server has memory 37 to store program instructions for the software application and a processor 39 that executes the instructions.

Each of the servers 35a-c can provide service to a finite number of clients at any given time. As demand increases, additional servers may be added to the server farm 35a-c. Rather than assigning clients to specific servers 35a-c, all clients access the servers 35a-c through load balancer 25 and switch 30, which are placed at the "front-end" of ASP 20.

Load balancer 25 has memory 27 that stores program instructions for a load-balancing program and a processor 29 that executes the instructions. Load balancer 25 has a unique network address (for example, an IP address) that is used by all

clients to access the services of ASP 20. Because clients access servers 35 through load balancer 25, ASP 20 can add additional servers 35 to the network, each with new unique network addresses, without the client having to be informed of the new unique network address.

When running latency-sensitive applications in packet-switched networks, it is preferable, and sometimes necessary, that once communication is established between the remote client computer and a particular server 35a, 35b or 35c, the communication is maintained through the same server until terminated by the client. The attribute of maintaining communication with the same server is known as "sticky." Also, the same server may need to be used because some protocols use more than one Port ID, which may be Transmission Control Protocol (TCP) ports, User Datagram Protocol (UDP) ports, or both. To prevent load balancer 25 from becoming a service bottleneck, an aspect of the invention provides that after one of servers 35a, 35b or 35c has been assigned, communications from the remote computers to the assigned server will bypass the load balancer 25 and proceed directly to the assigned server by addressing the message with the unique network address of the assigned server. This method allows ASP 20 to reduce the traffic through load balancer 25, increasing the quality of service for all of its clients.

To access the services of ASP 20, the remote computer L₁-L₆ may need to run a software application provided by the ASP 20 that is stored in the remote computer's memory 3.

Alternatively, the software application that is executed by the remote computers may be provided by other means. For example, the software application may be a game application purchased in shrink-wrapped packaging at a consumer electronics store. In this case, the software application is programmed to communicate with the ASP servers 35. The user need only configure the software application with the unique network address (e.g., an IP address or domain name) of the ASP 20, which will typically be the address of the load balancer 25.

The software application running on the client's remote computer may allow the client to see on a display a list of the applications that are available through the ASP 20. Further information beyond a list of the applications may also be displayed. For example, the client may see on a display a list of a plurality of "incarnations" of the applications. For example, in the context of gaming, there may be multiple different "Quake" game "worlds" running and "Diablo" game "worlds" running. If ASP 20 provides more than one application, the client may select the application that the client wants to use from a menu. By selecting an application, or other similar predetermined user command, the software on the remote computer

causes the remote computer to send a request for service, or a "probe request." The probe request is sent from the remote computer to the ASP 20 through NAT 15a or 15b.

FIG. 2 shows an embodiment of a method for establishing direct communications between a remote computer and an assigned server. First, the remote computer sends, at step 50, a probe request to the load balancer. The load balancer receives the probe request at step 55, and selects and assigns a server using a conventional selection algorithm. At step 60, the load balancer forwards the probe request to the assigned server. The assigned server receives the forwarded probe request at step 65. Next, the assigned server generates a "probe response" to be sent to the remote computer containing the unique IP address of the assigned server (e.g., 35b from FIG. 1) in the body of the response at step 70. The probe response is addressed to the NAT associated with the remote computer.

Alternatively, the probe response may be generated by the load balancer. In such a case, the load balancer may send two messages, the probe response to the remote computer and a message to the assigned server informing the server of the service request.

The remote computer receives the probe response at step 75, and extracts the unique network address of the assigned server (35b) from the response at step 80. The unique network address

is extracted by the software provided by the ASP on the remote computer without the client's knowledge. Using this unique address, the remote computer thereafter communicates directly with assigned server 35b at step 85, bypassing the load balancer. This prevents load balancer 25 from becoming a service bottleneck and avoids the latency problems and inefficient use associated with unnecessarily routing traffic through the load balancer.

FIG. 3 is a diagram of a probe request data packet 150 that may be sent by the remote computer to load balancer 25 to request service. Data packet 150 contains IP header 155 and UDP header 160. IP header 155 and UDP header 160 are address fields that are conventionally used in data packets that are transmitted over standard packet-switched networks. The IP header 155 contains the network address of the load balancer on the initial service request. On subsequent transmissions, the IP header 155 may contain the IP address of the assigned server.

After the load balancer or assigned server receives the data packet, the IP header and UDP header are removed and the remainder of data packet 150 is delivered to the port ID specified by UDP header 160. The port specifies a location within the server that responds to requests and communications of the particular software application provided by the ASP. In this embodiment, UDP messages are used. However, an equally

valid implementation may use TCP messages or any other appropriate data format. The remaining fields of data packet 150 contain information that is unique to the software application that is being provided by the ASP 20. The information provided by the remaining packet fields 165, 170, and 175 may permit direct communications between the remote computer and the assigned server.

Length field 165 indicates the total length of fields 165, 170, and 175. For example, the length field 165 may be 2 bytes, field 170 may be 2 bytes, and field 175 may be 4 bytes. In that case, length field 165 would indicate a total length of 8 bytes, as shown in FIG. 3. Type field 170 indicates to the remote computer and server 35 what type of data packet is being transmitted. In this embodiment, the remote computer and server 35 recognize that a packet is a probe request or response by the data "0xFFFF" in type field 170. However, any unique data set may be used. Address field 175 contains the IP address of the NAT associated with the remote computer sending the probe request. This allows the assigned server to send a probe response to the remote computer. In this embodiment, address field 175 is 32 bits in length. If subsequent versions of IP protocols are implemented, address field 175 may increase in size to 128 bits or more.

FIG. 4 is a diagram of a probe response data packet 180 that may be sent by the assigned server (or alternatively, by the load balancer) to the remote computer after the receipt of the probe request from the remote computer. The format of data packet 180 is similar to that of data packet 150. Data packet 180 contains IP header 185 and UDP header 190. IP header 185 contains the network address of the NAT associated with the remote computer. UDP header 190 contains the address of the specific port in the remote computer that responds to requests and communications of the particular software application provided by the ASP 20.

Length field 200 indicates the total length of fields 200, 205, and 210. Type field 205 indicates to the remote computer and assigned server the type of data packet being transmitted. In this embodiment, the remote computer and the assigned server recognize that a packet is a probe request or response by the data "0xFFFF" in type field 170. However, any unique data set may be used. Address field 210 contains the unique IP address of the assigned server sending the probe response. The inclusion of the real IP address of the assigned server allows the remote computer to bypass load balancer and communicate directly with the assigned server.

There are a number of server-based applications that transmit latency-sensitive messages. Examples of applications

include Internet telephony, multi-player gaming, music sharing, and other "peer-to-peer" type applications. In the example of Internet telephony, a client at computer L_1 may communicate with any other client at local computers L_2 - L_6 or a non-local client at computers N_1 - N_6 . The clients may communicate using Voice over Internet Protocol ("VoIP"). Using VoIP, data packets containing communications are transmitted by means of the Internet Protocol. The resulting communication has two-way full-duplex incoming and outgoing calling capability.

To communicate using VoIP, the client would send a request to ASP 20 for service. The request for service, i.e., probe request, is directed to the load balancer at the ASP. Using the method described in FIG. 2, the remote computer may establish direct communication with an assigned server at the ASP bypassing the load balancer. By establishing communication with a server (in the VoIP context this server is often called a gatekeeper or proxy), the ASP is notified that the client is available to place and receive calls. The client remains in communication with the assigned server until the client terminates the service session.

Other clients of the ASP may also establish direct communications with a server at an ASP. The other clients of the ASP may connect to servers at ASP 20, 21, or 23 as shown in FIG. 1. If the client places a call to a client that has

established communications with a server of ASP 20, the servers of ASP 20 will use a conventional method of negotiation to establish communications between the clients. If the other client has established communication with servers at ASP 21 or ASP 23, the servers of ASP 20 and the servers of ASP 21 or ASP 23 may communicate through Internet 40 using a conventional server-to-server protocol. The communication between the servers of ASP 20, 21, and 23 allow clients to transmit and receive messages in real time using VoIP.

In the example of multi-player gaming, a client of ASP 20 may interact in a gaming setting with other clients of ASP 20, ASP 21, or ASP 23 in real time. To set up the game, the client sends a probe request to the load balancer of the ASP. A client may establish communication with a server using the method described in FIG. 2.

Once connected to the assigned server, the client may enter the gaming application's "lobby". The gaming lobby describes the game that is available or in progress on the server, and may also provide a listing of other clients that are participating in the game. The client may be the only client of the ASP using the gaming application, or the client could be one of many users. In the example of multiple users, the gaming application tracks the activity of a particular client in the game and

transmits that information to all other clients that are playing the game.

Referring to FIG. 1, if the gaming application is running on a server at ASP 20, clients of ASP 21 may establish
 5 communication with a server of ASP 20 over Internet 40. Thus, a client in one locale may participate in a gaming application with a client in another locality in real time.

After direct communication has been established between a remote computer and an assigned server, the remote computer may
 10 monitor the communication using a fault tolerance technique in accordance with another aspect of the invention. This fault tolerance technique indicates whether the user computer and the assigned server are still in communication. Often, the client may realize that the communication has been terminated by the
 15 absence of activity or lack of a response by another client. In these situations, an indication from the fault tolerance technique may not be necessary. However, the fault tolerance technique may be helpful in applications where inactivity by another client is not immediately recognized.

20 FIG. 5 shows the process by which the fault tolerance technique monitors the communication between the remote computer and the assigned server. The connection may be monitored periodically (for example, once every 60 seconds or alternatively once after every tenth transmission) through the

transmission of messages between the remote computer and the assigned server. The frequency at which the connection is monitored may vary depending on the application.

Periodically during the communications between the remote
 5 computer and the assigned server, the software application on the remote computer determines that the connection between the remote computer and the assigned server should be verified. The remote computer then sends, at step 250, a probe request to the assigned server. This probe request may be similar to the
 10 request transmitted to begin a service session discussed previously, and may be of the format shown in FIG. 3.

If the connection between the remote computer and the assigned server is operative, the assigned server receives the probe request at step 260. At step 270, the assigned server
 15 generates and sends a probe response to the remote computer. The probe response may be in the format of data packet 180 shown in FIG. 4.

If the remote computer receives a probe response from the assigned server, then the remote computer is aware that the
 20 connection is still operative. In that case, the remote computer continues to transmit messages to the assigned server until the fault tolerance technique requires that another probe request be sent. If the remote computer does not receive a probe response after a certain period of time, then the remote

computer, and the client, knows there is a problem with the connection. In that case, the remote computer discontinues all traffic with the assigned server at step 280. The remote computer may then send a new probe request to the load balancer to establish a new communication at step 50 shown in FIG. 2.

This fault tolerance technique is an easy way to inform clients that they have lost connection. Informing clients that they have lost their connection at the earliest opportunity limits the frustration of clients who have lost the connection, and also minimizes the amount of data that may have been transmitted but lost after the connection failed.

A number of embodiments of the invention have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the invention. Accordingly, other embodiments are within the scope of the following claims.